УДК 004.65

НГО ТХАНЬ ХУНГ

МЕТОД ВЕРТИКАЛЬНОЙ КЛАСТЕРИЗАЦИИ ОТНОШЕНИЙ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

В статье представлена целевая функция для вертикальной кластеризации отношений в системах баз данных, построенная на основе нового оценочного критерия - вероятности кэш-попадания. Испытание полученной оценочной формулы с помощью программного моделирования показывает ее высокую точность. Кроме того, описан эвристический алгоритм, аппроксимирующий решение поиска наилучшей схемы кластеризации.

Ключевые слова: *вертикальная кластеризация отношений, целевая функция, оценочный критерий, вероятность кэш-попадания, эвристический алгоритм.*

Введение. Под вертикальной кластеризацией некоторого отношения мы понимаем декомпозицию без потерь его на два или более отношений, которые содержат непересекающиеся, за исключением ключевых атрибутов, подмножества атрибутов исходного отношения. В случае, если некоторое подмножество атрибутов чаше используются в запросах, чем подмножество остальных атрибутов, то эта техника может в среднем уменьшить объем данных, которые должны быть обработаны при заданном множестве запросов. В результате кластеризация приводит к повышению производительности кэш-системы в отдельности и всей системы в целом.

Как указано в [1], многие алгоритмы кластеризации были разработаны на основе статистической и образной классификации и анализа. Они группируют данные, используя различные критерии. Наиболее часто используемым критерием является квадратическая ошибка. Недостаток этих алгоритмов заключается в их трудоемкости и малоэффективности. Кроме того, их оценочные критерии не связаны ни с каким показателем системного исполнения, что вызывает некоторое недоверие к полученному результату. В [2] авторы разработали комплексный метод, в котором сначала выбираются потенциальные схемы кластеризации (варианты декомпозиции отношения). Затем эти схемы анализируются устройством оптимизации, построенным в данной системе управления базами данных (СУБД). После чего выбирается наилучшая схема для проведения кластеризации. Этот метод представляет большой интерес, но он слишком сложен для реализации. Кроме того, реализация устройства оптимизации существенно зависит от текущей настройки СУБД и от самого алгоритма оптимизации.

Постановка задачи. Принимая во внимание то, что существует тесная связь между кластеризацией и производительностью кэш-системы, необходимо сформулировать оценочный критерий для вертикальной кластеризации отношений вероятности кэш-попадания и разработать эффективный алгоритм поиска оптимальной схемы декомпозиции по этому критерию. Как и в большинстве алгоритмов в этой области, рассмотренных в [1], в качестве входных данных принимается матрица вероятностей использования атрибутов (Attribute Usage Matrix). Дополнительными входными данными

являются длины атрибутов (в байтах). Как и в [2], в данной работе мы будем считать, что число проекций декомпозиции равно двум, и обе они расположены в одной и той же табличной области.

Спецификации кэш-системы и системы баз данных. Рассматриваем некоторую базу данных (БД), отношения которой уже находятся в 5-й форме нормализации (5НФ) [3]. Пусть R – некоторое отношение, A – множество его атрибутов, $K \quad A$ - единственный потенциальный ключ, который является первичным ключом. Пусть $A \setminus K$ и, следовательно, отношение не является полностью ключевым.

Допустим, что каждый запрос использует подмножество атрибутов некоторого кортежа, определяемого его номер в таблице соответствующего отношения. Это означает, что запрос – последовательность трех операций: выбор отношения, выбор подмножества атрибутов и выбор кортежа. Поэтому запрос может быть определен как тройка $\left(R, \Delta, ID\right)$, где R - некоторое отношение, Δ - подмножество запрошенных атрибутов, ID - номер кортежа. Допустим, что выбор кортежей не зависит от выбора отношения, от выбора подмножества атрибутов и подчиняется равномерному закону. Пусть имеется Q различных запросов к базе данных. Статистическая информация о вероятностях их выполнения описывается значениями

$$p_1, p_2, \ldots, p_Q,$$

где p_i - вероятность вызова запроса q_i;

$$p_1 + p_2 + \ldots + p_Q = 1, \quad 0 \quad p_i \quad 1, \quad 1 \quad i \quad Q.$$

Относительно рассматриваемого отношения вероятности использования атрибутов описаны в табл.1, где величина u_{ij} принимает значение p_i, если запрос q_i требует атрибут a_j, и ноль, в противном случае.

Таблица 1

Матрица вероятностей использования атрибутов

Атрибуты Запросы	a_1	a_2	…	a_M
q_1	u_{11}	u_{12}	…	u_{1M}
q_2	u_{21}	u_{22}	…	u_{2M}
…	…	…	…	…
q_Q	u_{Q1}	u_{Q2}	…	u_{QM}

Пусть размер кэш-памяти для данного отношения ограничен длиной *L* байтов. Запись и чтение из кэша выполняются по кортежам. Кортеж загружается в кэш целиком, если, по крайней мере, один из его атрибутов востребован. В начале работы кэш-память пуста. Во время работы она заполняется кортежами. Для текущего запроса, если копия требуемого кортежа присутствует в кэше, тогда обращение к внешней памяти не происхо-

дит. В противном случае кортеж извлекается с диска, и его копия размещается в часть кэш-памяти, выделенная для данного отношения (если кэш-память не выделяется для данного отношения, то ее кортежи не будут кэшироваться вообще). В первом случае, мы говорим, что было кэш-попадание, а во втором – кэш-промах. После того как кэш-память заполняется, новый кортеж должен вытеснить один из имеющихся в нем кортежей. Так как период, пока кэш-память не заполнена, является переходным, то целевую функцию построим без учета этого периода. Будем считать, что стратегия замены, реализуемая в данной СУБД, является рандомизацией, т.е. выбор позиции, где хранится замещаемый при очередном промахе кортеж, подчиняется равномерному закону распределения вероятностей.

Описанные предположения сделаны в результате дедуктивного анализа литературы [1], [2], [3], [4]. Заметим, что эти предположения, за исключением сведения о вероятностях использования атрибутов, сформулированы только с целью математических выводов и испытания достоверности предполагаемого оценочного критерия, а не для требования к области его применения.

Метод реализации вертикальной кластеризации отношений. Рассмотрим вертикальную кластеризацию отношения R на два отношения R_1 и R_2. Обозначим множества атрибутов этих отношений A, A_1, A_2 соответственно. Кластеризация реализуется так, что верны следующие соотношения:

$$A_1 = A_2 \quad A, \; A_1 = A_2 \quad K, \; |A_1| > |K|. \qquad (1)$$

Используя метод имитации вертикальной кластеризации, создадим для отношений R_1 и R_2 постоянные таблицы T_1 и T_2 соответственно. При этом номера кортежей таблицы T сохраняются для соответствующих кортежей подтаблиц T_1, T_2. Запросы, обращающиеся к исходному отношению R, будут переписаны, чтобы реализовать обращения к отношениям R_1 и R_2. Если допустить, что любой запрос содержит, по крайней мере, один из неключевых атрибутов, тогда некоторый исходный запрос (R, Δ, ID) заменяется только по одному из трех вариантов:

1) одним запросом к отношению R_1, (R_1, Δ, ID), если все атрибуты исходного запроса присутствуют в A_1, т.е. $(\Delta \quad A_1)$;
2) одним запросом к отношению R_2, (R_2, Δ, ID), если все атрибуты исходного запроса присутствуют в A_2, т.е. $(\Delta \quad A_2)$;
3) двумя запросами $(R_1, \Delta \quad A_1, ID)$ и $(R_2, \Delta \quad (A_2 \setminus K), ID)$ одновременно в тех случаях, когда не ключевые атрибуты подмножества Δ частично присутствуют и в A_1, и в A_2, т.е. $((\Delta_j \quad A_1) \quad) \; ((\Delta_j \quad A_2) \quad K)$.

Кэш-память, отводимая для кэширования R, теперь разделена на две части для кэширования отношений R_1 и R_2. Размеры этих частей являются параметрами кластеризации.

Вывод оценочного критерия. Для получения критерия, который оценивает вероятность кэш-попадания для кэширования исходного отношения и его проекций, ведём следующие обозначения:

$U = \{u_{ij}\}$ – матрица вероятностей использования атрибутов отношения R;

N – число кортежей отношений R, R_1, R_2;

M – число атрибутов отношения R;

p_i – вероятность того, что к СУБД обращается *i*-й запрос, $i = 1, 2, \ldots, Q$;

$\Delta_i \quad A$ – подмножества атрибутов, которые востребованы *i*-ым запросом;

l_j – длина *j*-го атрибута отношения R, $j = 1, 2, \ldots, M$;

l_R, l_{R1}, l_{R2} – длины кортежей соответствующих отношений R, R_1, R_2;

L_R, L_{R1}, L_{R2} – размеры частей кэш-памяти, отведенных для кэширования отношений R, R_1, R_2, где $L_{R1} + L_{R2} = L_R$;

N_R, N_{R1}, N_{R2} – количества позиций кэш-памяти, где хранятся кортежи соответствующих отношений R, R_1, R_2;

оператор $[\ ]$ – оператор получения целой части вещественного числа.

Вероятность кэш–попадания для кэширования исходного отношения. Обращение к кэш-памяти будет успешным (кэш-попадание) только в том случае, если востребованный кортеж является одним из N_R кортежей, копия которых уже присутствуют в кэше. Так как выборы кортежей подчиняются равномерному закону и не зависят друг от друга, то вероятность кэш-попадания одна и та же для каждого запроса и равна:

$$P_R = \frac{N_R}{N}, \qquad (2)$$

где $N_R = \mathbf{min}\ \ N, \ \frac{L_R}{l_R} \ = \mathbf{min}\ \ N, \ \frac{L_R}{\sum\limits_{\forall j: a_j \ A} l_j}$.

Вероятность кэш-попадания для кэширования проекций-отношений. Обозначим случаи, когда исходный запрос заменен по первому, второму, третьему вариантам событиями C, C_1, C_2 соответственно; случай,

когда поиск исходного требуемого кортежа в кэше завершается успешно событием B. Согласно формуле полной вероятности вероятность кэш-попадания оценивается как:

$$P_{R_1R_2} = \sum_{v=1}^{3} \left(p\left(B / C_v \right) \cdot p\left(C_v \right) \right). \qquad (3)$$

Для v=1 исходный запрос $\left(R,\Delta ,ID \right)$ заменяется запросом $\left(R_1,\Delta ,ID \right)$. Поэтому поиск кортежа, определяемого двойкой $\left(R, ID \right)$, завершится успешно тогда и только тогда, когда поиск кортежа, определяемого двойкой $\left(R_1, ID \right)$, завершится успешно. Аналогично случаю, рассмотренному в (2), имеем следующее выражение:

$$p\left(B / C_1 \right) = \frac{N_{R1}}{N}, \qquad (4)$$

где $N_{R1} = \mathbf{min}\left(N, \frac{L_{R1}}{l_{R1}} \right) = \mathbf{min}\left(N, \frac{L_{R1}}{\sum_{\forall j: a_j \in A_1} l_j} \right)$.

Из условия замены исходного запроса очевидно следующее выражение:

$$p\left(C_1 \right) = \sum_{\forall i: \Delta_i \subseteq A_1} p_i . \qquad (5)$$

Для v = 2, аналогично предыдущему обсуждению, получим:

$$p\left(B / C_2 \right) = \frac{N_{R2}}{N}, \qquad (6)$$

где $N_{R2} = \mathbf{min}\left(N, \frac{L_{R2}}{l_{R2}} \right) = \mathbf{min}\left(N, \frac{L_{R2}}{\sum_{\forall j: a_j \in A_2} l_j} \right)$;

$$p\left(C_2 \right) = \sum_{\forall i: \Delta_i \subseteq A_2} p_i . \qquad (7)$$

В случае $v = 3$ исходный запрос $\left(R,\Delta ,ID \right)$ заменяется двумя запросами $\left(R_1,\Delta \cap A_1,ID \right)$ и $\left(R_2,\Delta \cap \left(A_2 \setminus K \right),ID \right)$ одновременно, и поиск кортежа $\left(R, ID \right)$ завершится успешно тогда и только тогда, когда оба поиска кортежей $\left(R_1, ID \right)$ и $\left(R_2, ID \right)$ завершатся успешно одновременно. Таким образом, верно следующее:

$$p\left(B / C_3 \right) = p\left(B_{R1} \right) \cdot p\left(B_{R2} \right) = \frac{N_{R1}}{N} \cdot \frac{N_{R2}}{N};$$

$$p\left(B / C_3 \right) = \frac{N_{R1} \ N_{R2}}{N^2} . \qquad (8)$$

Из свойства полной группы вероятность наступления события *C₃* вычисляется как:

$$p\left(C_3 \right) = 1 - p\left(C_1 \right) - p\left(C_2 \right) . \qquad (9)$$

В итоге вероятность кэш-попадания в случае кэширования проекций-отношений выглядит следующим образом:

$$P_{R_1 R_2} = \frac{N_{R1}}{N} \ p\left(C_1 \right) + \frac{N_{R2}}{N} \ p\left(C2 \right) + \frac{N_{R1} \ N_{R2}}{N^2} \ p\left(C_3 \right) . \qquad (10)$$

Обозначим:

$$a = \frac{p\left(C_1 \right)}{N}; \quad b = \frac{p\left(C_2 \right)}{N}; \quad c = \frac{p\left(C_3 \right)}{N^2} .$$

Формулу (9) можно переписать следующим образом:

$$P_{R_1 R_2} = a \ N_{R1} + b \ N_{R2} + c \ N_{R1} \ N_{R2} . \qquad (11)$$

Целевая функция для поиска наилучшей схемы кластеризации отношений. Для проведения кластеризации необходимо решить следующую задачу (будем называть задачей (*)):

Пусть даны: множество атрибутов A, количество кортежей N, матрица U, длины атрибутов l_j и размер кэш-памяти L_R.

Найти наилучшую схему для проведения кластеризации данного отношения.

Любая схема кластеризации полностью идентифицируется четырьмя параметрами A_1, A_2, N_{R1}, N_{R2}. Качество схем будем оценивать формулой (11). Чем больше значение оценочного критерия (11), тем выше производительность кэш-системы. Поэтому наилучшая схема кластеризации является решением задачи дискретного программирования (в дальнейшем будем называть ее задачей (**)):

$$\mathbf{P}\left(A_1, A_2, N_{R1}, N_{R2} \right) = a \ N_{R1} + b \ N_{R2} + c \ N_{R1} \ N_{R2} \rightarrow \text{max}. \qquad (12)$$

С ограничениями:

$$A_1 = A_2 \quad A, \ A_1 = A_2 \quad K, \ \left| A_1 \right| > \left| K \right|; \qquad (13)$$

$$N_{R1} + l_{R1} \quad N_{R2} \ l_{R2} \quad L_R ; \qquad (14)$$

$$N_{R1} \quad 0, \ N_{R2} \quad 0; \qquad (15)$$

$$N_{R1}, \ N_{R2} \text{ – целые}. \qquad (16)$$

Так как предлагаемый оценочный критерий не трудоемок для вычисления, задача (**) может быть решена с помощью алгоритма полного перебора для входных данных с небольшой размерностью. При больших значениях величины M и интервалов варьирования N_{R1}, N_{R2} необходимо разработать более эффективный алгоритм.

Алгоритм поиска наилучшей схемы полным перебором. Задачу дискретного программирования (**) можно решить методом полного

перебора [5]. Сначала перебираем все допустимые пары $\left(A_1, A_2\right)$ по условию (13), потом для каждого из найденных вариантов перебираем все допустимые пары $\left(N_{R1}, N_{R2}\right)$ по условиям (14), (15) и (16). Заметим, что при фиксации значения A_1, что следует из условия (13), значение A_2 автоматически фиксируется. По (13) имеем $\left(2^{M-|K|}-1\right)$ допустимых пар значений $\left(A_1, A_2\right)$. При фиксации значения N_{R1}, значение N_{R2}, которое обеспечивает максимум целевой функции (12), должно быть определено выражением

$$N_{R2} = \mathbf{min}\ \ N, \ \frac{L_R - N_{R1} l_{R1}}{l_{R2}} \quad .$$

Значит после фиксации пары $\left(A_1, A_2\right)$ имеем $\mathbf{min}\ \ N, \ \frac{L_R}{l_{R1}} \ +1$ допустимых пар $\left(N_{R1}, N_{R2}\right)$, потенциально обеспечивающих максимум (4). Таким образом, алгоритм полного перебора должен реализовать не менее $\left(2^{M-|K|}-1\right) \ \mathbf{min}\ \ N, \ \frac{L_R}{l_R} \ +1$ вычислений формулы (12).

Например, рассмотрим отношение R с шестью атрибутами, где a_1 - ключевой атрибут. Длины атрибутов - 30, 37, 51, 21, 46, 11 (байт). Матрица вероятностей использования атрибутов (табл.2).

Таблица 2

Пример матрицы вероятностей использования атрибутов

Атрибуты / Запросы	a_1	a_2	a_3	a_4	a_5	a_6
q_1	.194	0	.194	.194	0	.194
q_2	.089	.089	0	0	0	.089
q_3	.281	.281	0	0	.281	0
q_4	.157	.157	0	0	.157	.157
q_5	.279	0	0	0	.279	.279

Размер кэш-памяти, выделенной для данного отношения, равен 2000 (байт). Число кортежей равно 200.

Для этого примера наилучшая схема кластеров:

$A_1 = \{a_1, a_2, a_5, a_6\}, \quad A_2 = \{a_1, a_3, a_4\}$, $N_{R1} = 16,\ N_{R2} = 0$.

С этой схемой вероятность кэш-попадания, рассчитанная по формуле (12), равна 0.06448, и она возросла на 1.29 (раз) относительно случаю без кластеризации, вероятность кэш-попадания при котором равна 0.05.

Программное моделирование предложенного метода кластеризации. Испытание достоверности предлагаемого оценочного критерия было выполнено с помощью программного моделирования. Данное программное средство реализовано на Delphi и предоставляет следующие возможности:

- генерировать случайные длины атрибутов рассматриваемого отношения и матрицу вероятностей их использования;
- решить задачу (**) полным перебором;
- генерировать случайные потоки запросов, соответствующие генерированной матрице вероятностей использования атрибутов;
- моделировать работу кэш-системы с возможностью выбора той или иной стратегии замещения (было реализовано 4 стратегий: рандомизация, Clock, LRU и LFU [6]);
- моделировать системы кластеризации отношений, как описано выше;
- вычислить средние значения экспериментальных кэш-попаданий и их дисперсию.

Для каждого набора входных данных для задачи (*) производили поиск наилучшей схемы алгоритмом полного перебора. Статистическим моделированием были получены производительности кэш-системы при 1000 потоках запросов, длина каждого потока равна 5000 запросов. Во всех этих случаях относительная ошибка экспериментального среднего от теоретического значения по формуле (13) составила менее 2.5%. При моделировании других стратегий был получен аналогичный результат.

Эвристический алгоритм, аппроксимирующий решения задачи (**). Исследуя свойства решений задачи (**), полученных алгоритмом полного перебора, было замечено, что почти во всех случаях либо $N_{R1} = \min\ N, \frac{L_R}{l_{R1}}$, либо $K_2 = \min\ N, \frac{L_R}{l_{R2}}$, т.е. почти вся часть кэш-памяти, отведенная для исходного отношения, передается на кэширование только одной из проекций-отношений. Из данного факта мы предполагаем кэшировать только отношение R_1 из схемы кластеризации, а другое отношение совсем не кэшируем. Таким образом, будем аппроксимировать решение задачи (**) решением задачи (***):

$$P(A_1, A_2, N_{R1}, N_{R2}) = a\ N_{R1} = {}_{\forall i: \Delta_i \subset A_1}\ p_i\ \frac{L_{R1}}{{}_{\forall j: a_j\ A_1}\ l_j} \to \max. \quad (17)$$

С ограничениями:

$$A_1 = A_2 \quad A,\ A_1 = A_2 \quad K,\ |A_1| > |K|; \quad (18)$$

$$N_{R1} = \min \ \ N, \ \frac{L_R}{l_{R1}} \quad ; \tag{19}$$

$$N_{R2} = 0 . \tag{20}$$

Эвристический алгоритм находит множество A_1 путём последовательного отбора его атрибутов. Вначале $A_1 = A$. На каждой итерации будет удален из подмножества A_1 атрибут $a_j \quad K$ с длиной l_j, оценка принадлежности к подмножеству A_1 которого принимает наименьшее значение. Эта оценка вычисляется по формуле:

$$PU_j = \underset{\forall i: a_j \ \ \Delta_i}{\ } p_i \Big/ l_j \qquad \left(1 \quad i \quad Q, 1 \quad j \quad M\right). \tag{21}$$

После удаления атрибута a_j будут обнуляться все элементы i-й строки матрицы U, где $u_{ij} > 0$. На данной же итерации из множества A_1 удаляются также все атрибуты $a_r \quad A_1 \ \left(a_r \quad K\right)$, где $\underset{\forall i: a_r \ \ \Delta_i}{\ } p_i = 0$. Полученное при данной итерации разбиение $\left(A_1, \left(A \setminus A_1\right) \quad K\right)$ является одним из интересуемых вариантов разбиения. Итерационный процесс повторяется до тех пор, пока $\left|A_1\right| > \left|K\right|$. Вычисляется оценочный критерий (17) для разбиения $\left(A, K\right)$ и для всех разбиений, полученных во всех итерациях. Наилучшая схема разбиения из них является решением задачи (***).

Таким образом, данный алгоритм реализует не более $\left(\left|A\right| - \left|K\right| + 1\right)$ итераций, поэтому он во много раз эффективнее, чем алгоритм полного перебора.

Для анализа эффективности эвристического алгоритма к разработанному программному средству был добавлен модуль, который его реализует. С помощью полученной программы было произведено 5000 экспериментов. В каждом эксперименте были сгенерированы входные данные задачи (*). Затем решена задача (*) с помощью алгоритма полного перебора и эвристического алгоритма. Анализ результатов показывает, что эвристический алгоритм даёт одну и ту же схему кластеризации, что и алгоритм полного перебора в более чем 90% экспериментов. Для класса эвристических алгоритмов [7] такой процент совпадений решений, на наш взгляд, является достаточно приемлемым.

Выводы. В данной работе получена целевая функция для вертикальной кластеризации с новым оценочным критерием – вероятность кэш-попадания, а также выполнено испытание достоверности полученной формулы с помощью программного моделирования. Ее простота и высокая точность подтверждают перспективность предложенного метода вертикальной кластеризации отношений при проектировании и реконструкции баз данных с

целью повышения системной производительности. Кроме того, разработан эвристический алгоритм, аппроксимирующий решение задачи поиска наилучшей схемы кластеризации. Его эффективность и точность позволяют существенно сократить время проведения кластеризации.

Библиографический список

1. Sharma Chakravarthy, Jaykumar Muthuraj, Ravi Varadarajan, Shamkant B. Navathe. An Objective Function for Vertically Partitioning Relations in Distributed Databases and its Analysis. In *Distributed and Parallel Databases* 2(2): 183-207(1994).

2. Sanjay Agrawal, Vivek Narasayya, Beverly Yang. Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design. In SIGMOD 2004, June, 2004.

3. К. Дж. Дейт. Введение в системы баз данных. 7-е издание /К.Дж.Дейт. – М., СПб, Киев: Вильямс, 2001.

4. Вильям Пэйдж, Дэвид Остин, Виллард Берд II, Николас Чейз и др. Специальное издание: Использование Oracle8/8I. Вильямс, 2000.

5. Ковалев М.М. Дискретная оптимизация – Целочисленное программирование /М.М.Ковалев.– М.: УРСС, 2003.

6. Таненбаум Э. Современные операционные системы /Э.Таненбаум. – СПб: Питер, 2004.

7. Natallia Kokash. An introduction to heuristic algorithms. 2005**//** http://dit.unitn.it/~kokash/documents/Heuristic_algorithms.pdf

Материал поступил в редакцию 1.08.08.

NGO THANH HUNG

VERTICAL PARTITIONING RELATIONS METHOD IN RELATIONAL DATABASE

In this paper we introduce a new objective function for vertical partitioning relations in relational databases. It has been built with the new evaluative criterion: cache hit probability. Also present heuristic algorithm for finding the optimal partitioning schema.

Index terms: vertical partitioning, objective function, evaluation criterion, cache hit probability, heuristic algorithm.

НГО Тхань Хунг (р. 1980), аспирант кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» ДГТУ по научной специальности 05.13.01 «Системный анализ, управление и обработка информации» (2005-2008). Окончил ДГТУ (2005) по специальности «Программное обеспечение вычислительной техники и автоматизированных систем».

Научные интересы: методы повышения производительности кэш-систем и систем управления распределенными базами данных.

Автор 7 научных работ.